**Amendments to the Specification:**

Please replace the title of the invention with the following amended title:

~~METHOD FOR INTERFACING COMPONENTS~~ METHODS FOR INTERFACING COMPONENTS AND FOR WRITING AND READING A SERIAL DATA STREAM TO AND FROM A COMPONENT ASSOCIATED WITH A PARALLEL DATA STREAM

Please replace the paragraph on Page 1, starting at line 3 which starts with "This invention relates generally" with the following amended paragraph:

This invention relates generally to the field of interfacing components and, more particularly, to an intelligent module for interfacing an ~~IDE~~ integrated device electronics (IDE) hard disk drive with an ~~MFM-based~~ modified frequency modulation (MFM)-based control system.

A2

Please replace the paragraph on Page 1, starting at line 6 which starts with "In the mid-1980's" with the following amended paragraph:

In the mid-1980's, hard disk drives (herein-after referred to as hard disks) often utilized electrical interfaces compliant with the Institute of Electrical and Electronics Engineers (IEEE) ~~IEEE~~ 412 standard. Such hard disks are often referred to as "dumb" devices, as they must be told what to do and when to do it (e.g., by a controller). In such a system, physical media is remote from controller electronics, meaning the interface break lies between the mechanical portion of the drive (i.e., the magnetic media, stepper motors, heads, etc.) and the controller electronics.

A3

Please replace the paragraph on Page 3, starting at line 3 which starts with "According to one embodiment" with the following amended paragraph:

According to one embodiment of the present invention, a method for interfacing components is provided. The steps of the method include receiving from a first component a request for an access operation comprising one of at least a write operation and a read operation, ~~and~~ the request comprising a requested data address associated with a second component, and determining whether data corresponding to the requested data address is missing from memory, the memory being independent of the second component. When the data is missing from the memory, the method includes the steps of intercepting the request, loading the data from the second component to the memory, and performing the requested access operation. According to such an embodiment, modified data is converted from a first communication type to a second communication type and written to the memory and to the second component during a write operation, and data is converted from the second

A4

communication type to the first communication type and read by the first component during a read operation.

Please replace the paragraph on Page 3, starting at line 16 which starts with "In another embodiment" with the following amended paragraph:

AS

In another embodiment of the present invention, a method for writing a serial data stream to a component associated with parallel data streams is provided. This method includes the steps of receiving a serial data stream, converting the serial data stream to a parallel data stream, loading the parallel data stream in memory, and writing the parallel data stream from the memory to a component. Yet another embodiment of the present invention is directed towards a method for reading a serial data stream from a component associated with parallel data streams. ~~A~~ The method according to this embodiment includes the steps of receiving a request for data associated with the component, receiving a parallel data stream from memory in response to the request for the data, converting the parallel data stream to a serial data stream, and outputting the serial data stream.

Please replace the paragraph on Page 3, starting at line 26 which starts with "Still other aspects of the" with the following amended paragraph:

A6

Still other aspects of the present invention will become apparent to those skilled in this art from the following description wherein there is shown and described various embodiments of this invention, simply by way of illustration. As will be realized, the invention is capable of other different aspects and embodiments without departing from the scope of the invention. Accordingly, these exemplary objects are not intended to, nor do they, limit the scope of the present invention in any way, and the drawings and descriptions should be regarded as illustrative in nature and not as restrictive in nature. Further preferred embodiments of the present invention involve a computer readable medium including instructions capable of performing the steps of the various methods of the present invention.

Please replace the paragraph on Page 4, starting at line 19 which starts with "Figure 6 is an" with the following amended paragraph:

~~Figure 6 is an illustration of~~ Figures 6A-6C illustrate a flowchart describing an interrupt subroutine according to one embodiment of the present invention;

Please replace the paragraph on Page 4, starting at line 23 which starts with "Figure 8 is an" with the following amended paragraph:

~~Figure 8 is an illustration of~~ Figures 8A-8B illustrate a flowchart describing an interrupt subroutine according to one embodiment of the present invention;

Please replace the paragraph on Page 5, starting at line 1 which starts with "Figure 11 is an" with the following amended paragraph:

~~Figure 11 is an illustration of~~ Figures 11A-11C illustrate a flowchart describing an interrupt subroutine according to one embodiment of the present invention;

Please replace the paragraph on Page 5, starting at line 3 which starts with "Figure 12 is an" with the following amended paragraph:

~~Figure 12 is an illustration of~~ Figures 12A-12C illustrate a flowchart describing an interrupt subroutine according to one embodiment of the present invention;

Please replace the paragraph on Page 5, starting at line 11 which starts with "Referring now to the drawings" with the following amended paragraph:

Referring now to the drawings in detail, wherein like numerals indicate the same elements throughout the views, Figures 1 and 2 depict an interface between system 20 and interface module 30, and interface module 30 and drive 40 (~~referred to~~ shown collectively in Figure 2 as drive module 35), respectively, according to an exemplary embodiment of the present invention, which utilizes an MFM-based controller and an IDE hard drive. According to the exemplary embodiment illustrated herein, to system 20, interface module 30 emulates an interface of an MFM drive (i.e., it appears as/emulates physical media - e.g., a literal spinning disk with heads floating over a media surface, with stepper motor, cylinder selection, and head and drive selects - to the system). Furthermore, according to such an

embodiment, all relevant data and control lines of system 20 can be handled by interface module 30.

Please replace the paragraph on Page 6, starting at line 12 which starts with "With respect to inputs received" with the following amended paragraph:

With respect to inputs received from system 20, interface module 30 can accept the following signals: head select 0, head select 1, head select 2; drive select 0, drive select 1; step; direction; MFM write gate, and MFM write data. The three head selects associated with an exemplary embodiment of the present invention are used to form a binary number that can be used to specify one of, for example, eight simulated heads that are to be active on the next read or write operation. Meanwhile, the drive select lines can be used to specify, for example, a drive that system 20 needs to access. Referring now to the step signal, this input can be used to trigger a motor (of an MFM drive) to step read/write heads one track in the direction determined by the state of the direction input.

A12

Please replace the paragraph on Page 6, starting at line 21 which starts with "With respect to the direction input" with the following amended paragraph:

With respect to the direction input, this signal can be used to indicate the direction heads are to ~~spin~~ move. For example, when the line is set low, the input can represent when the heads are to move towards a spindle hub (inside tracks), while when the line is set high, the input signal indicates that the heads are to move toward the outer tracks. In addition, the write gate input can be used to indicate whether a write operation or a read operation is to be performed. For example, an active low signal can be asserted low for a write operation, and asserted high for a read operation. Finally, with respect to data input, two differential data lines, +MFM write and -MFM write can be used to transmit MFM data.

A13

Please replace the paragraph on Page 8, starting at line 3 which starts with "Among other tasks" with the following amended paragraph:

Among other tasks, processor 22 can fetch, decode, and/or execute instructions, and/or transfer information to and from other resources, such as memory 24, and/or management module 32, such as by way of a bus. Furthermore, processor 22 can comprise

A14

an on-board microprocessor. For example, processor 22 can comprise an Intel® 80C186XL running at 25 ~~MHZ~~ megahertz (MHZ).

---

Please replace the paragraph on Page 8, starting at line 15 which starts with "Referring now to Figure 4" with the following amended paragraph:

---

A15

Referring now to Figure 4, according to one embodiment of the present invention, local RAM 26A comprises two 128Kx8 static RAM ("SRAM") chips, addressed in the illustrated embodiment at the beginning of the address range of processor 22 (e.g., 0 – 3FFFFh). Meanwhile, ROM 26B can comprise a 64K upper block of programmable read-only memory ("PROM"), which can be erasable (i.e., an "EPROM"). EPROM or ROM 26B can, for example, comprise two 27C256 chips, providing 64K bytes of ROM that is addressed in the upper 64K block (e.g., F0000h-FFFFFh) of the address range of processor 22.

---

Please replace the paragraph on Page 8, starting at line 23 which starts with "According to an exemplary" with the following amended paragraph:

---

A16

According to an exemplary embodiment of the present invention, cache 28 comprises cylinder RAM that is divided into blocks (e.g., four 128K byte blocks, respectively labeled A, B, C, D). Reference numeral 28 may be referred to hereafter as cache 28 or cylinder RAM 28. Within each block, cylinder RAM 28 can be segmented into pages (e.g., four pages), where each page can hold the equivalent of one track of data (e.g., 32K for MFM data). In one embodiment, at least an entire track is loaded into cylinder RAM 28 so that, once the seek signal is released, an entire emulated "rotation" of the media can be continuously presented to system 20. Although, in an exemplary embodiment, cylinder block A is used for cylinder 0 data, which typically stores a software chunk table, and cylinder blocks B, C, and D hold the data of the last three cylinders requested by system 20, all of the cylinder blocks could also be freely cached.

---

Please replace the paragraph on Page 9, starting at line 3 which starts with "Storing data corresponding to" with the following amended paragraph:

---

A17

Storing data corresponding to four cylinders of drive 40 in cache 28 can, for example, allow for efficient use of disk caching, which can reduce the number of accesses to the drive.

For example, ~~and~~ as will be discussed below, during a read operation requested by system 20, data cached in cylinder RAM 28 can be read from the cylinder RAM with no accesses to drive 40. In one embodiment, accesses to drive 40 will only be required if the requested cylinder data is not in cylinder RAM 28 (i.e., a cache "miss") or if a write to the drive is required.

---

Please replace the paragraph on Page 9, starting at line 10 which starts with "Referring now to Figure 5" with the following amended paragraph:

---

Referring now to Figure 5, a simplified diagram of the architecture of cache 28, according to one embodiment of interface module 30, is shown. According to an exemplary embodiment of the present invention, after system 20 has issued a cylinder seek (e.g., issuing a series of head step pulses while drive 40 is selected), comparators (see below for discussion regarding cylinder RAM arbitration blocks 58 and 62 in Fig. 9) can be used to determine if the requested cylinder address is present in cylinder RAM 28 (i.e., a cache "hit"). In the instance of a cache hit, according to one embodiment, system 20 does not have to wait for processor 22 to fetch the requested track as the system is granted immediate access to the data already found in cache 28.

---

Please replace the paragraph on Page 9, starting at line 19 which starts with "Meanwhile, if there is" with the following amended paragraph:

---

Meanwhile, if there is a cache miss, management module 32 can generate an update cache interrupt (see also discussion below regarding cylinder RAM arbitration blocks 58 and 62 in Fig. 9) that is communicated to processor 22, such as by way of interrupt request line INT0, which initiates a first interrupt subroutine, such as that depicted in Figure 6. In general, with the initiation of the first subroutine, processor 22 can detain system 20, such as by causing the de-assertion of "seek-complete" (a handshake signal used to "hold off" the system and interface module 30 during lengthy emulation tasks), until the requested cylinder data can be loaded from drive 40 to cylinder RAM 28, for example. Cache arbitration block 64 can be used to keep track of data in cylinder RAM 28.

---

Please replace the paragraph on Page 10, starting at line 3 which starts with "Referring now to the exemplary" with the following amended paragraph:

Referring now to the exemplary embodiment shown in ~~Figure 6~~ Figures 6A-6C, after unmasking the IDE drive interrupt at step 602, the seek-complete control line is held off, as shown in step 604. According to the illustrated embodiment, when reading the requested cylinder data to cylinder RAM 28, processor 22 overwrites the oldest/least used cylinder block/page in the cylinder RAM. For example, processor 22 can determine which block in cylinder RAM 28 is oldest by inspecting a cache status latch or port associated with management module 32, and can set a pointer to the respective corresponding data address. Processor 22 can then read a write status word from a track dirty status register provided by management module 32 to determine which tracks within the oldest cache block have been modified in cylinder RAM 28, but not modified on drive 40 (i.e. which tracks are "dirty"), as shown in step 606. Processor 22 then writes these dirty tracks to drive 40.

A21

Please replace the paragraph on Page 10, starting at line 14 which starts with "According to the illustrated embodiment" with the following amended paragraph:

According to the illustrated embodiment, this write operation involves determining whether any of the cylinders are dirty, as shown in steps 608, 614, 622, and 630. If a cylinder is dirty, the respective current cylinder address is determined (see, e.g., steps 616, 624, and 632) and the respective dirty cylinder tracks are written to drive 40 (see, e.g., steps 610, 618, 626, and 634). At such a write operation, a drive write ~~LED~~ light emitting diode (LED) can also be flashed, as shown in steps 612, 620, 628, and 636.

A22

Please replace the paragraph on Page 11, starting at line 1 which starts with "In addition, a second interrupt request" with the following amended paragraph:

In addition, a second interrupt request line INT1 associated with processor 22 can be in electrical communication with, for example, a host interrupt request line of the drive (e.g., WINTRQ of drive 40). As shown in Fig. 7, this interrupt, referred to hereinafter as the drive interrupt, can be used to inform processor 22 that drive 40 has completed a requested operation. For example, the drive status register can be read as shown in step 700. The drive interrupt can also be used to initiate diagnostics on drive 40, such as to check for any errors with the drive and/or its operation, as shown in steps 702, 704, 706, 708, and 710. For example, after the drive status register is read as shown in step 700, the processor 22

A23

determines if a drive diagnostic command has been issued as shown in step 702. If the drive diagnostic command has been issued, the processor 22 reads the error register as shown in step 704. The processor 22 determines if the register error bit is set as shown in step 706. If the register error bit set is set, the processor 22 reads the error register as shown in step 708. If the processor 22 determines the register error bit is not set, the processor sets "dd.error" equal to zero as shown in step 710.

A23

---

Please replace the paragraph on Page 12, starting at line 3 which starts with "For example, as shown in the exemplary" with the following amended paragraph:

---

For example, as shown in the exemplary embodiment depicted in ~~Figure 8~~ Figures 8A-8B, after unmasking the IDE drive interrupt and masking the update cache interrupt at step 800, the seek-complete control line is held off, as shown in step 802. According to the illustrated embodiment, processor 22 reads a write status word from a track dirty status register provided by management module 32 to determine which tracks within the oldest cache block have been modified in cylinder RAM 28, but not modified on drive 40 (i.e. which tracks are "dirty"), as shown in step 804. Processor 22 then writes these dirty tracks to drive 40.

A24

---

Please replace the paragraph on Page 12, starting at line 23 which starts with "Although, as can be understood" with the following amended paragraph:

---

Although, as can be understood by one of ordinary skill in the art, management module 32 can be, for example, implemented in software, or by using discrete logic (e.g., transistor-transistor logic (TTL)~~TTL~~, complimentary metal-oxide semiconductor (CMOS)~~CMOS~~, etc.) or custom / semi-custom devices (e.g., application-specific integrated circuits or ASICs), and can comprise a single component or a plurality of components, the management module of the illustrated embodiment comprises a single programmable logic device ("PLD"), such as a field programmable gate array ("FPGA"). A PLD from the Flex® 10K family of devices offered by ~~the~~ Altera Corporation of San Jose, California, such as the Flex® 10K20, can be used to form management module 32. As can be understood by one of ordinary skill in the art, one advantage of using a Flex® 10K20 PLD can include the utilization of embedded array blocks (EABs), which can be ideal for RAM, ROM, and first in, first out ("FIFO") functions.

A25

---

Please replace the paragraph on Page 13, starting at line 18 which starts with "Meanwhile, a mux-decode block" with the following amended paragraph:

---

Meanwhile, a mux-decode block 44 can be used to steer data to and/or from appropriate ports in interface module 30. For example, mux-decode block 44 can be used to determine which channel (e.g., Cylinder Y RAM Select, Cylinder X RAM Select, Track Dirty Status Register, Cylinder B Address, Requested Cylinder Port, Read Status Port 1, Cylinder C Address, Cylinder D Address, Cache Status, and/or Read Status Port 2) is selected for assertion on a multiplexed processor address/data bus associated with an exemplary embodiment of the present invention. As shown in Table 1 below, mux-decode block 44 can utilize combinatorial logic to decode signal lines (e.g., processor address lines A0, A1, A2, and A3, and an address select pin IO from processor 22 that denotes input/output space is selected~~, IO~~) and memory bank selects for cache 28 (CYL-X and CYL-Y), to determine which channel is to be selected. Accordingly, processor 22 can access the selected port.

A26

---

Please replace the paragraph on Page 15, starting at line 4 which starts with "As the exemplary processor" with the following amended paragraph:

---

As the exemplary processor 22 multiplexes its address and data bus, local-bus-arbitration block 50 is used to separate address lines from the multiplexed processor address/data bus, which can be used to communicate between interface module 30, processor 22, local RAM 26A, EPROM 26B and Cache 28 (Fig. 3). Block 50 can also provide decoding circuitry for cache 28. For example, gated latches can be used to latch address lines from the bus when an address latch enable ("ALE") signal goes active. In an exemplary embodiment, combinatorial logic is used to decode addresses in the ranges of, for example, 00000-3FFFFH and F0000H-FFFFFH. When an address does not fall within this range, it is considered as being associated with cache 28 (e.g., 40000-BFFFFH).

A27

---

Please replace the paragraph on Page 18, starting at line 3 which starts with "Once processor 22" with the following amended paragraph:

---

Once processor 22 writes new data into the cylinder RAM block/page that contained the oldest data, it can update the current cylinder (e.g., B, C, or D) address latch or port. This

A28

port can be used to tell a magnitude comparator (such as ~~a~~ an 11-bit magnitude comparator block 60) and/or processor 22 the cylinder address of the data stored in that area of cylinder RAM 28. As previously discussed, in one embodiment of the invention, the data stored in cylinder A is always cylinder 0 data, while the data stored in cylinders B, C, and D is variable. Moreover, a bit (e.g., track-0) can be activated when the step count has been decremented to be zero (i.e., when the cylinder count bits are zero), indicating the "simulated" heads are now located over the first cylinder on the media (e.g., simulating a situation where a MFM drive is at track 0). Furthermore, output(s) (e.g., WR-CYL-B, WR-CYL-C, and WR-CYL-D) of magnitude comparator(s) 60 can be used as chip selects to the cylinder RAM chips, thereby enabling cache 28 as appropriate.

A28

---

Please replace the paragraph on Page 18, starting at line 3 which starts with "Once processor 22" with the following amended paragraph:

---

Referring now in more detail to 11-bit magnitude comparator blocks 60, these blocks can be used to compare the requested cylinder address (e.g., from the requested cylinder address port) with the cylinder addresses of the data in the blocks of cylinder RAM 28. In one embodiment, blocks 60 comprise, for example, a simple ~~XNOR~~ exclusive NOR (XNOR) and 12-IN NAND circuit with gate input.

A29

---

Please replace the paragraph on Page 19, starting at line 3 which starts with "Referring now to channel management block" with the following amended paragraph:

---

Referring now to channel management block 66, this block can coordinate additional blocks, such as those involving functions ~~such as~~ to synchronize data, control write channel, control read channel, and control the address counter. In such an embodiment, system 20 can control read and write channels of interface module 30 through channel manager block 66. For example, during a write operation associated with the highlighted example involving an MFM control system and IDE hard drive, serial data from system 20 can be routed into a write channel block 70, where it is converted to parallel data. After, for example, every eighth bit, the write channel function can increment the address in an address counter, such as with address counter block 68.

A30

---

Please replace the paragraph on Page 19, starting at line 24 which starts with "For example, an 18-bit" with the following amended paragraph:

---

For example, an 18-bit cylinder RAM address can be generated from four 74191 counters. A 74191 counter is a 4-bit binary up/down counter with asynchronous load. According to one embodiment of the present invention, as shown in Table 3, the counters are used to determine bits 0-14, while HEAD0, 1, and 2 from system 20 are used to determine bits 15-17. An index-pulse (see previous discussion with respect to ~~dig~~ digital-one shots block 48), such as an end-of-track pulse, for example, can be generated at the end of each track block (e.g., A0-A14 = 22,000d), where the HEAD bits determine which track (e.g., 0, 1, 2, 3) is selected within the current cylinder RAM block. In one embodiment, only A0-A14 are automatically incremented by the read and write channels of interface module 30.

A31

---

Please replace the 2nd paragraph on Page 20, which starts with "Once the terminal count" with the following amended paragraph:

---

Once the terminal count (e.g., sixteen bits) is reached, the parallel data output from shift register 74 can be latched into two octal flip-flops. From here, a series of flip-flops and gates can be used to meet any RAM write cycle timing requirements (e.g., ~~100ns~~ 100 nanoseconds (ns)). In an exemplary embodiment, write channel ~~boek~~ block 70 is active when the drive-select, ready, seek-complete, and write-gate signals of system 20 are active.

A32

---

Please replace the paragraph on Page 21, starting at line 19 which starts with "A write-chan-FF-low block" with the following amended paragraph:

---

A write-chan-FF-low block 76 can comprise a low-byte latch that can be used to capture part of the parallel data from serial-to-parallel-shift-register 74. A reset on this latch can be used to default the output to 00h. Meanwhile, a ~~write-chan-FF-hi~~ write-chan-FF-high block 78 can comprise a ~~hi-byte~~ high-byte latch that can be used to capture another part of the parallel data from serial-to-parallel-shift-register 74. A reset on this latch can also be used to default the output to 00h.

A33

---

Please replace the paragraph on Page 22, starting at line 24 which starts with "A read-chan-FF-low block" with the following amended paragraph:

A read-chan-FF-low block 84 can comprise a low-byte latch that can be used to capture part of the parallel data from cylinder RAM 28. A reset on this latch can be used to default the output to 24h. Meanwhile, a ~~read-chan-FF-hi~~ read-chan-FF-high block 86 can comprise a ~~hi-byte~~ high-byte latch that can be used to capture another part of the parallel data from cylinder RAM 28. A reset on this latch can be used to default the output to 95h.

A34

Please replace the paragraph on Page 22, starting at line 29 which starts with "Block 88 can organize" with the following amended paragraph:

Block 88 can ~~organize one or more of the following blocks into one sheet~~ coordinate additional blocks, such as IDE data bus block 90; Cylinder RAM block92; LED Status block 94; and Status Port block 96. ~~IDE data bus~~ IDE data bus block 90, for example, can be used to control the transceivers that interface drive 40. In one embodiment of the invention, block 90 comprises a combinatorial circuit that can decode an IDE drive command.

A35

Please replace the paragraph on Page 23, starting at line 26 which starts with "Starting with block" with the following amended paragraph:

Starting with block 112, ~~after~~ where cylinder 0 of drive 40 is loaded into cylinder RAM block A, the read channel is "kick-started," as previously discussed and shown in step 114. The count bits corresponding to cache 28 are set such that the blocks of cylinder RAM are given different orders of priority for filling, and then a seek-complete line is released and a drive-ready signal is sent to system 20, each respectively being shown in steps 116 and 118. Finally, resuming with step 120, the update cache interrupt is unmasked and, as shown in step 122, the drive select interrupt is unmasked.

A36

Please replace the paragraph on Page 24, starting at line 12 which starts with "As shown in Figure 11" with the following amended paragraph:

As shown in ~~Figure 11~~ Figures 11A-11C, when a cache miss has occurred the corresponding light emitting diode (e.g., a READ LED), is turned on as shown in block 210. As shown in block 212, a determination is then made if cylinder 0 is requested (i.e., if at track 0). If cylinder 0 was not requested, a determination is then made as to which cache block is

A37

oldest (e.g., B, C, or D), as shown by blocks 214, 216 and 218. Once the appropriate block is identified, the corresponding memory address is set, as shown in blocks 220, 222, 224 and 226, and the requested data is loaded into the appropriate block, as identified in steps 228-246. For example, disk drive head is set to zero (0) as shown in step 228. The disk drive sector is set to equal one (1) as shown in step 230. The Multiple Mode and Call Windrv is set as shown in step 232. A determination is performed to determine if the disk drive is in error as shown in step 248. If there is a disk drive error, the error is displayed. If there is not a disk drive error, a command for reading Multiple Mode and Call Windrv is performed. Another determination if there is a disk drive error is performed as shown in step 250. If there is not a disk drive error, a command is performed to increment memory address = 200h by the number of sectors per block as shown in step 236. A determination of the sector is performed as shown in step 238. If sector is less than or equal to 45, the disk drive sector is incremented by number of sectors per block as shown in step 240. If the disk drive sector is greater than 45, a determination is performed to determine if the disk drive head is less than or equal to 3 as shown in step 242. If the disk drive head is greater than 3, address segment is incremented by 800h as shown in step 244 and the disk drive head is incremented as shown in step 246. The read routine can also includes error checking steps 248 and 250 (Fig. 11B), which can be used to switch from multi-sector mode to single sector mode and read in all undamaged sectors if an error occurs while reading in a sector. Finally, at step 252 (*Fig. 11C), the READ LED can be turned off.

A37

---

Please replace the paragraph on Page 24, starting at line 23 which starts with "Meanwhile, with respect to a write" with the following amended paragraph:

---

Meanwhile, with respect to a write operation, as previously discussed, such a routine is primarily invoked during a drive update interrupt and can be used to bring the IDE hard disk 40 into agreement with cache 28 during idle periods. One routine for performing the write operation is depicted in ~~Figure 12~~ Figures 12A-12C. As shown in ~~this figure~~ these Figures, a WRITE LED can be turned on upon initiation of this routine, as shown in block 310. As shown in steps 312-324, the appropriate cylinder RAM pointer is set. For example, a determination of the disk drive cylinder is performed as shown in step 312. If the disk drive cylinder is not equal to zero, a determination of the cylinder ram address is preformed as shown in step 314. If the cylinder ram address does not equal 60,000h, the cylinder ram address is compared against 80,000h. as shown in step 316. If the cylinder ram address does

A38

not equal 80,000h, the cylinder ram address is set to A0,000h as shown in step 318. A determination is made of the disk drive head as shown in step 320. If the disk drive head is equal to 1, 800h is added to the cylinder ram offset as shown in step 322. The segment is set to equal the cylinder ram address plus the disk drive head multiplied by 1000h. As further shown in steps 326-342, the cylinder RAM can be copied to the IDE hard disk 40. For example, the disk drive sector is set to equal one (1) as shown in step 326. The Multiple Mode and Call Windrv is set as shown in step 328. A determination is performed to determine if the disk drive is in error as shown in step 344. If there is a disk drive error, the error is displayed. If there is not a disk drive error, a command for reading Multiple Mode and Call Windrv is performed as shown in step 330. Another determination if there is a disk drive error is performed as shown in step 346. If there is not a disk drive error, a command is performed to increment memory address = 200h by the number of sectors per block as shown in step 332. A determination of the disk drive sector is performed as shown in step 334. If disk drive sector is less than or equal to 45, the disk drive sector is incremented by number of sectors per block as shown in step 336. If the disk drive sector is greater than 45, a determination is performed to determine if the disk drive head is less than or equal to 3 as shown in step 338. If the disk drive head is greater than 3, address segment is incremented by 800h as shown in step 340 and the disk drive head is incremented as shown in step 342. Moreover, as with the read routine, error checking can be done, such as that shown in blocks 344 and 346 (Fig. 12B). Finally, as shown in step 348 (Fig. 12C), the WRITE LED can be turned off upon completion of the operation.

A38

---

Please replace the paragraph on Page 25, starting at line 15 which starts with "Also according to one embodiment" with the following amended paragraph:

---

Also according to one embodiment of the invention, interface module 30 is designed to tolerate operating temperatures from at least about 0 °C to about 65 °C, and storage temperatures from at least about -40 °C to about 100 °C . Humidity of about 5% to about 95% non-condensing should also be acceptable. Moreover, the rated temperature range of drive 40 should be ~~at least~~ about 5 °C to about 55 °C operating, and about –40 °C to about 60 °C non-operating.

A39

---

Please replace the paragraph on Page 25, starting at line 26 which starts with "In one embodiment of the " with the following amended paragraph:

A40

In one embodiment of the present invention, a flexible disk interface is also controlled by system 20. Accordingly, interface module 30 can also interface a floppy drive, such as through back-plane connector 23, a data/control connector ~~(J1)~~, and a power connector ~~(J2)~~. Such a flexible disk drive should be compatible with the American National Standards Institute ("ANSI") 3.80-1981 specification. In addition, the form factor can be the industry-standard 3.5 inch drive.